Quasi-Monte Carlo Methods for Some Linear Algebra Problems

Aneta Karaivanova* Michael Mascagni†

First proposed by von Neumann and Ulam, Monte Carlo methods (MCMs) for solving linear algebra problems have been known since the middle of the last century. They give statistical estimates for the elements of the inverse of a matrix or for components of the solution vector of a linear system by performing random sampling of a certain chance variable whose expected value is the desired solution. Perhaps the first application of MCMs in linear algebra appeared in a paper by Forsythe and Leibler [11] in 1950. In the following years significant contributions were made, especially by Wasow [20], Curtiss [5], Halton [14], Hammersley and Handscomb [13] and Sobol' [19]. These methods were recognized as useful in the following situations [21]: when obtaining a quick rough estimate of solution, which will then be refined by other methods; when the problem is too large or too intricate for any other treatment; when just one component of the solution vector or one element of the inverse matrix is desired.

There has been renewed interest in MCMs in recent times, for example [1, 2, 6, 7, 8, 9, 10, 15], the primary reason for this is the efficiency of parallel MCMs in the presence of high communication costs. The second reason for the recent interest in MCMs is that the methods have evolved significantly since the early days. Much of the effort in the development of Monte Carlo methods has been in the construction of variance reduction techniques which speed up the computation by reducing the the rate of convergence of crude MCM, which is $O(N^{-1/2})$. An alternative approach to acceleration is to change the type of random sequence, and hence improve the behavior with N. **Quasi-Monte Carlo methods** (QMCMs) use quasirandom (also known as low-discrepancy) sequences instead of pseudorandom sequences, with the resulting convergence rate for numerical integration being as good as $O((\log N)^k N^{-1})$. The first results of using QMCMs for linear algebra problems were presented by Mascagni and Karaivanova (see for example [16, 17]).

Quasi-Monte Carlo methods often include standard approaches for variance reduction. The fundamental feature underlying all QMCMs, however, is the use of a quasirandom sequence. In this paper the convergence and the complexity of quasi-Monte Carlo methods for estimating the solution of systems of linear algebraic equations (SLAE), inverting of matrices and finding extremal eigenvalues are studied when quasirandom sequences are used. An error bound for computing matrix-vector product is established. Numerical experiments with large sparse matrices are performed using different quasirandom number (QRN) sequences. The results indicate that for all of the considered problems improvements in both the magnitude of the error and the convergence rate can be achieved using QRNs in place of pseudorandom numbers.

1 The Problems

Given a matrix $B = \{b_{ij}\}_{i,j=1}^{n}$, $B \in \mathbb{R}^n \times \mathbb{R}^n$, and a vector $b = (b_1, \ldots, f_n)^t \in \mathbb{R}^n$ consider the following three problems:

Problem 1. Evaluating the inner product

$$J(u) = (h, u) = \sum_{i=1}^{n} h_i u_i \tag{1}$$

of the solution $u \in \mathbb{R}^n$ of the linear algebraic system $Bu = b$ and a given vector $h \in \mathbb{R}^n$.

After choosing a matrix $M \in \mathbb{R}^n \times \mathbb{R}^n$ such that $MB = I - A$, [3], where $I \in \mathbb{R}^n \times \mathbb{R}^n$ is the identity matrix, and $Mb = f$, $f \in \mathbb{R}^n$, the matrix equation $Bu = b$ becomes

$$u = Au + f. \tag{2}$$

Assumming the matrices M and A are both nonsingular, and $|\lambda(A)| < 1$ for all eigenvalues $\lambda(A)$ of

*Department of Computer Science, Florida State University, Tallahassee, FL, **USA**, *AND* Bulgarian Academy of Sciences, CLPP, Sofia, **Bulgaria**, E-mail: aneta@csit.fsu.edu, anet@copern.bas.bg

†Department of Computer Science and School of Comp. Science and Inf. Technology, Florida State University, Tallahassee, FL, **USA**, E-mail: mascagni@cs.fsu.edu, URL: http://www.cs.fsu.edu/~mascagni

A, then the general stationary linear iteration

$$u^{(k+1)} = Au^{(k)} + f$$

may be used for solving the system $Bx = b$.

Problem 2. Inverting matrices, i.e., computing the matrix

$$C = B^{-1}, \tag{3}$$

where $B \in \mathbb{R}^n \times \mathbb{R}^n$ is a given real matrix.

Assumming the matrix B is non-singular and $||\lambda(B)| - 1| < 1$ for all eigenvalues $\lambda(B)$ of B, we construct the matrix $A = I - B$. Then the inverse matrix $C = B^{-1}$ can be presented as $C = \sum_{i=0}^{\infty} A^i$ and the desired approximation of C is the truncated series with the corresponding truncation error.

Problem 3. Evaluating extremal eigenvalues:

$$Au = \lambda(A)u. \tag{4}$$

It is assumed that the matrix A is non-singular and $\lambda_{min} = \lambda_n < \lambda_{n-1} \leq \lambda_{n-2} \leq \ldots \leq \lambda_2 < \lambda_1 = \lambda_{max}$.

We consider the matrix A and also its *resolvent* matrix $R_q = [I - qA]^{-1} \in \mathbb{R}^{n \times n}$. The eigenvalues of the matrices R_q and A are thus connected by the equation $\mu = \frac{1}{1-q\lambda}$, and the eigenvectors of the two matrices coincide.

The largest eigenvalue can be obtained using the power method applied to the matrix A [8]:

$$\lambda_{max} = \lim_{i \to \infty} \frac{(h, A^i f)}{(h, A^{i-1} f)}, \tag{5}$$

or using the power method applied to the resolvent matrix [9]:

$$\mu^{(m)} = \frac{(h, [I - qA]^{-m} f)}{(h, [I - qA]^{-(m-1)} f)} \underset{m \to \infty}{\longrightarrow}$$

$$\mu_{max} = \frac{1}{1 - q\lambda} \text{ for } q > 0. \tag{6}$$

For computing the smallest eigenvalue, we use the fact that for negative values of q, the largest eigenvalue, μ_{max}, of R_q corresponds to the smallest eigenvalue λ_{min} of the matrix A, so we use (6) with $q < 1$.

2 MCMs for Linear Algebra

To solve these problems via MCMs (see, for example, [13, 19]) one has to construct for each problem a random process with mean equal to the solution of the desired problem. All of these methods are based on computing matrix-vector product.

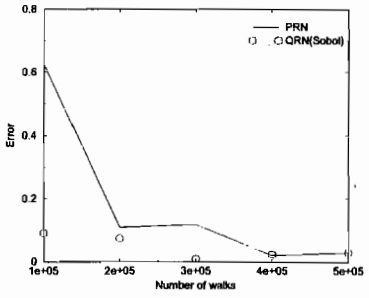

Figure 1: *MC and QMC: Accuracy versus number of walks for computing (h, x), where x is the solution of a system with 2000 equations.*

2.1 Matrix-Vector Product

Given a matrix A and vectors $f, h \in R^n$, we want to compute $h^T A^i f$ for some i, using a Monte Carlo method. Consider the following Markov chain $k_0 \to k_1 \to \ldots \to k_i$, where $k_j = 1, 2, \ldots, n$ for $j = 1, \ldots, i$ are natural numbers. The rules for constructing the chain are $P(k_0 = \alpha) = p_\alpha, P(k_j = \beta | k_{j-1} = \alpha) = p_{\alpha\beta}$ where p_α is the probability that the chain starts in state α and $p_{\alpha\beta}$ is the transition probability from state α to state β . Probabilities $p_{\alpha\beta}$ define a transition matrix P. We require that $\sum_{\alpha=1}^{n} p_\alpha = 1$ and $\sum_{\beta=1}^{n} p_{\alpha\beta} = 1$ for any $\alpha = 1, 2, ..., n$, and that the distribution $(p_1, ..., p_n)^t$ is permissible for the vector h and similarly the distribution $p_{\alpha\beta}$ is permissible for A [19]. Common constructions are to choose $p_\alpha = \frac{1}{n}$, $p_{\alpha\beta} = \frac{1}{n}$ which corresponds to *crude* Monte Carlo, or to choose $p_\alpha = \frac{|h_\alpha|}{\sum_{\alpha=1}^{n} |h_\alpha|}$; $p_{\alpha\beta} = \frac{|a_{\alpha\beta}|}{\sum_{\beta=1}^{n} |a_{\alpha\beta}|}$, $\alpha = 1, \ldots, n$. corresponds to *importance sampling* algorithms for matrix computations–the zero elements will never be visited and the elements with larger magnitude will be visited more often during the random walks on the elements of the matrix.

Now define weights for our Markov chain using the following recursion formula:

$$W_0 = 1, \; W_j = W_{j-1} \frac{a_{k_{j-1} k_j}}{p_{k_{j-1} k_j}}, \; j = 1, \ldots, i. \tag{7}$$

Following [19], it is easy to show that

$$E\left[\frac{h_{k_0}}{p_{k_0}} W_i f_{k_i}\right] = (h, A^i f), \; i = 1, 2, \ldots. \tag{8}$$

2.2 Monte Carlo estimations

To solve **Problem 1**, define the random variables

$$\theta[h] = \frac{h_{k_0}}{p_{k_0}} \sum_{j=0}^{\infty} W_j f_{k_j}. \quad (9)$$

It is known [19] that the mathematical expectation of this random variable is $E[\theta[h]] = (h, u)$.
The partial sum corresponding to (9) is defined as $\theta_i[h] = \frac{h_{k_0}}{p_{k_0}} \sum_{j=0}^{i} W_j f_{k_j}$. Thus the Monte Carlo estimate for (h, x) is

$$(h, x) \approx \frac{1}{N} \sum_{s=1}^{N} \sum_{i=0}^{l_s} \left(\frac{h_{k_0}}{p_{k_0}} W_i f_{k_i} \right)_s$$

where N is the number of chains and $\theta_i[h]_s$ is the value of $\theta_i[h]$ taken over the s-th chain. This estimate has a statistical error of size $O(\mathrm{Var}(\theta_i)^{1/2} N^{-1/2})$.

If we try to solve **Problem 2**, i.e., we want to compute the element $c_{rr'}$ of the matrix inverse to A, then we use the following equation [19]:

$$c_{rr'} = E\{ \sum_{i|k_i=r'} W_i \}, \quad (10)$$

where $(i|k_i = r')$ means a summation only for weights W_i for which $k_i = r'$ and $C = \{c_{rr'}\}_{r,r'=1}^{n}$. The Monte Carlo estimate then is

$$c_{rr'} \approx \frac{1}{N} \sum_{s=1}^{N} \left[\sum_{(j|k_j=r')} W_j \right]_s.$$

To solve **Problem 3**, we use the equation (8) and also [9]

$$E[\sum_{i=0}^{\infty} q^i C_{i+m-1}^{i} \frac{h_{k_0}}{p_{k_0}} W_i f(x_i)] = (h, [I - qA]^{-m} f),$$

for $m = 1, 2, \ldots$, which allow us to express the estimates (5) and (6) as

$$\lambda_{max} \approx \frac{E[W_i f_{k_i}]}{E[W_{i-1} f_{k_{i-1}}]} \quad (11)$$

and

$$\lambda \approx \frac{E[\sum_{i=1}^{\infty} q^{i-1} C_{i+m-2}^{i-1} W_i f(x_i)]}{E[\sum_{i=0}^{\infty} q^i C_{i+m-1}^{i} W_i f(x_i)]}. \quad (12)$$

We use MCM for an approximate calculation of these expected values:

$$\lambda_{max} \approx \frac{\sum_{s=1}^{N} (W_m f_{k_m})_s}{\sum_{s=1}^{N} (W_{m-1} f_{k_{m-1}})_s}$$

$$\lambda_{min} \approx \frac{\sum_{s=1}^{N} ([\sum_{i=0}^{l} q^i C_{i+m-1}^{i-1} W_{i+1} f(x_{i+1})])_s}{\sum_{s=1}^{N} ([\sum_{i=0}^{l} q^i C_{i+m-1}^{i} W_i f(x_i)])_s},$$

$q < 1$.

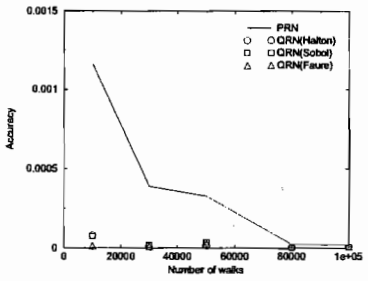

Figure 2: *MC and QMC: Accuracy versus number of walks for computing one component, x_{64}, of the solution for a system with 1024 equations.*

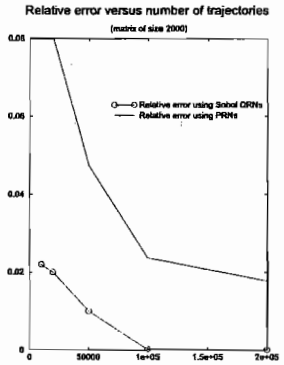

Figure 3: *Relative errors for λ_{max} using power MCM and quasi-MCM with different number of Markov chains for a sparse matrix 2000 × 2000.*

3 QMCMs for Matrix Computations

The methods presented here are based on computing $h^T A^i f$, and computing this scalar product is equivalent to computing an $(i+1)$-dimensional integral. Thus we may analyze using QRNs in this case with bounds from numerical integration. We do not know A^i explicitly, but we do know A and can use a random walk on the elements of the matrix to compute approximately $h^T A^i f$.

Using the following procedure: $G = [0, n)$, $G_i = [i-1, i)$, $i = 1, \ldots, n$, $f(x) = f_i$, $x \in G_i, i = 1, \ldots, n$, $a(x, y) = a_{ij}, x \in G_i, y \in G_j$, $i, j = 1, \ldots, n$, $h(x) = h_i$, $x \in G_i, i = 1, \ldots, n$, we can consider computing $h^T A^i u$ to be equivalent to computing a $(i+1)$-dimensional integral. Consider the scalar product $h^T A f$ bearing in mind that the vectors h, f, and the matrix A are normalized with factors of $1/\sqrt{n}$ and $1/n$ respectively and denoted

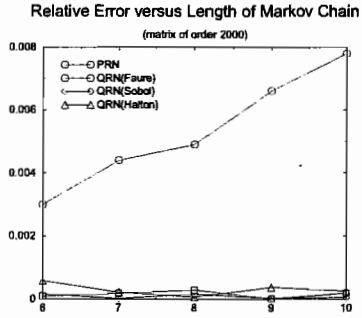

Figure 4: *Relative errors for λ_{max} using resolvent method with different length of Markov chains for a sparse matrix 2000 × 2000.*

by h_N, A_N, and f_N. In this case we have

$$|h_N^T A_N^l f_N - \frac{1}{N}\sum_{s=1}^{N} h(x_s)a(x_s,y_s)\dots a(z_s,w_s)f(w_s)|$$

$$\leq |h|^T |A|^l |f|.D_N^*.$$

Why are we interested in quasi-MCMs for the eigenvalue problem? Because the computational complexity of QMCMs is bounded by $O(lN)$ where N is the number of chains, and l is the mathematical expectation of the length of the Markov chains, both of which are independent of matrix size n. This makes QMCMs very efficient for large, sparse, problems, for which deterministic methods are not computationally efficient. Numerical tests were performed on general sparse matrices using PRNs and Sobol, Halton and Faure QRNs. Some results are shown in Figures 1, 2, 3, and 4.

References

[1] V. ALEXANDROV, Efficient parallel Monte Carlo methods for matrix computations, *Math. and Comp. in Simulation*, **47**:113-122, 1998.

[2] V. ALEXANDROV, A. KARAIVANOVA, "Parallel Monte Carlo Algorithms for Sparse SLAE using MPI", *Lecture Notes in Computer Science*, Springer, (J.Dongarra, E. Luque, T. Margalef, Eds.), vol. **1697**: 283-290, 1999.

[3] R. L. BURDEN, J. D. FAIRES, *Numerical Analysis*, Fifth Edition, Brooks/Cole Publishing Company, California, 1996.

[4] R. CAFLISCH, "Monte Carlo and quasi-Monte Carlo methods," *Acta Numerica*, **7**: 1–49, 1998.

[5] J. H. CURTISS, Monte Carlo methods for the iteration of linear operators, *J. of Math. Physics*, **32**, 1954, pp. 209-323.

[6] D. DANILOV, S. ERMAKOV, J. H. HALTON, Asymptotic complexity of Monte Carlo methods for solving linear systems, *J. of Stat. Planning and Inference*, **85**:5-18, 2000.

[7] I. DIMOV, V. ALEXANDROV, A. KARAIVANOVA, Resolvent Monte Carlo Methods for Linear Algebra Problems, *Math. and Comp. in Simulations*, Vol. bf. 55, 2001, pp. 25-36.

[8] I. DIMOV, A. KARAIVANOVA, "Iterative Monte Carlo algorithms for linear algebra problems", *LNCS*, Springer, **1196**: 66-77, 1996.

[9] I. DIMOV, A. KARAIVANOVA, "Parallel computations of eigenvalues based on a Monte Carlo approach", *Monte Carlo Methods and Applications*, Vol. **4**, Num.1: 33-52, 1998.

[10] B. FATHI, B. LIU, V, ALEXANDROV, "Mixed Monte Carlo Parallel Algorithms for Matrix Computation", *LNCS*, Springer, **2330**: 609-618, 2002.

[11] G. FORSYTHE, R. LEIBLER, Matrix Inversion by a Monte Carlo Method, *Math. Tables and Other Aids to Computation*, **4**:127-147, 1950.

[12] G. H. GOLUB, C.F. VAN LOON, *Matrix computations*, Johns Hopkins Univ. Press, Baltimore, 1996.

[13] J. HAMMERSLEY, D. HANDSCOMB, *Monte Carlo methods*, John Wiley & Sons, New York, London, Sydney, 1964.

[14] J. H. HALTON, Sequential Monte Carlo, *Proceedings of the Cambridge Philosophical Society*, 58 part **1**:57-78, 1962.

[15] J. H. HALTON, Sequential Monte Carlo Techniques for the Solution of Linear Systems, *SIAM J. of Sci. Comp.*, Vol.**9**, pp. 213-257, 1994.

[16] M. MASCAGNI, A. KARAIVANOVA, Matrix Computations Using Quasirandom Sequences, *LNCS*, Vol.**1988**, Springer, 2001, pp. 552-559.

[17] M. MASCAGNI, A. KARAIVANOVA, A Parallel Quasi-MCM for Computing Extremal Eigenvalues, *MCQMCMs 2000*, Springer: 369-380, 2002.

[18] H. NIEDERREITER, *Random number generation and quasi-Monte Carlo methods*, SIAM: Philadelphia, 1992.

[19] I. SOBOL', *Monte Carlo numerical methods*, Nauka, Moscow, 1973 (in Russian).

[20] W. WASOW, A note on the inversion of matrices by random walks, *Math. Tables and Other Aids to Computation*, **6**:78-78, 1952.

[21] J. WESTLAKE, *A Handbook of Numerical Matrix Inversion and Solution of Linear Equations*, J. Wiley & Sons, New York, 1968.